Development, Integration, and Implementation of the CDF Data Acquisition System for Run II at the Tevatron

B. Badgett[1],*F. Chlebana*[1], *A. Meyer*[1], *S. Nahn*[2], *C. Gay*[2], *Y. Guo*[1], *J. Patrick*[1], *L. Piccoli*[1] *S. Vejcik*[1], *M. Vittone*[1], *M. Votava*[1],

Fermi National Accelerator Laboratory, Batavia, IL, USA

Abstract

The CDF experiment at the Tevatron is undergoing an extensive set of upgrades to its abilities. During the coming collider run, CDF is expected to trigger and record events from ppbar collisions corresponding to instantaneous luminosities of $10^{32}cm^{-2}sec^{-1}$ with primary collisions occuring every 132 nsec and typical event sizes of 200 Kbytes. The implementation of a data acquisition system capable of efficiently recording this data is the most critical element of the experimental design. A presentation of the system pertaining to its developmental, integration, and commisioning stages is made. Details are discussed with regards to the structure of the software used in a system with widely dispersed processing nodes, messaging technologies, and designs for monitoring of the system performance.

Keywords: Data Acquisition,Colliders,control systems

1 Introduction

The Collider Detector at Fermilab (CDF) is undergoing an extensive upgrade of its capabilities for operations at the Fermilab Tevatron in 2001. The goals of the experiment are to record data from $p\bar{p}$ collisions at $\sqrt{s} = 1.8$ TeV for several years. Data is recorded from a dozen different detector systems triggered by a 2-level trigger system and filtered by software functioning as a 3rd trigger level. The basic construction of the detector system continues an evolution of the detector since 1987. The tracking, calorimeter, muon detection, and silicon vertexing systems provide over 700,000 channels. Data records are expected to be of order 200 kBytes per event taken at rates of 100 Hz. ppbar collisions will take place at rates of up to 1/132 nsec. The trigger system is designed to efficiently determine which events should be recorded with a goal of no dead-time during the Level 1 processing and less than 5processing. Data recorded by the detector elements will be digitized in VME crates distributed mostly close to the detector itself. The Data Acquisition system is responsible for delivering the data to and through a large ATM switching network and into the farm of processors operating the Level 3 filter software. Each VME crate contains a processing unit (Motorola MVME- 2603) and a collection of PCB boards designed and constructed by collaborators on the experiment.

The implementation of the system represents the culmination of many design decisions for the development and integration aspects of the project as well as for the final project.

2 Development

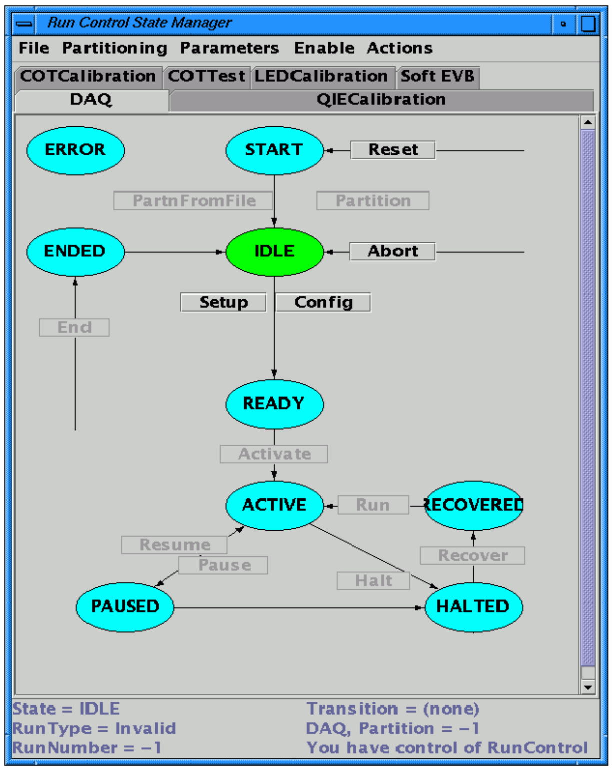
The implementation of the system represents the culmination of many design decisions for the development and integration aspects of the project as well as for the final project. During the development phase, a software environment is required that can control the operation of sets of VME-based boards. It is highly desireable that the environment be useful both at remote sites as well as in the CDF experimental hall. Nearly every board is required to function in a cooperative fashion with other independently designed systems so that some uniformity of software controls is desireable. Finally, it was considered beneficial to adopt a system that would allow for re-use of as many software elements as possible in the subsequent scaling of systems to the fully experiment. The development tools were designed around a distributed object model based on CORBA specifications and using TCP/IP for the communications between user controls on a host computer (either a UNIX-type or Windows-NT platform) and the processors in VME crates. The design uses a client-server protocol with a server operating on the VME processors. Client software is written in Java and server software in c. The VME processors use a Unix-like operating system, VxWorks. Users design interfaces for control of individual boards and subsequently produce Java packages and C-libraries as implementations. The C-libraries contain the atomic functionality needed to interact with the boards. At CDF, an implementation of the VITA VME standards, FISION, is used to access board elements from the VME code. Performance is adequate, greatly enhanced by good code management tools that allowed for portability between locations. Most importantly, since the basic board access environment (VxWorks and FISION) are the same as thouse used during the run, the extensive libraries of board-level code can be re-used in subsequent levels of development.

3 Integration and Implementation

Prior to operation of the full Data Acquisition System, a distinct integration phase is needed during which electronics and software is brought together at the experimental hall to test their abilities to interact appropriately. The Data Acquisition System is designed to be partitionable so that subsets of VME crates can be independently operated. The critical element of this system is the Trigger System Interface, a set of VME board designed to implement the communication protocols between the Trigger System and Detector VME crates. Because the flow of data from the crates is driven by the the Trigger System, the partitionability of the system is set by that of the TSI. For Run II, the CDF DAQ system will handle up to 8 partitions. These separate partitions are critical for implementing many different subsystems at the same time and which share many common resources. All subsystems bring their hardware and development software to Fermilab and rely on the teststand environmnents to check basic interactions in chains, or slices of the experiment. The next level of integration requires the coordinated sequencing of operations between boards and validating their response to signals from the Trigger System.

The software facility designed to coordinate activity among the different crates is called Run Control. It is comprised of a set of cooperating Java-based facilities that are responsible for initiating and receiving communications with VME crates and non-VME based software components (such as monitoring systems). Run Control is designed in a modular fashion so that subsystems can be used and tested during the integration phase and a natural path exists to evolve toward the final system capabilities. The sequencing of data-taking operations is the minimal capability and is accomplished by managing state transitions for each partition corresponding to a finite state diagram that describes the particular data-taking functionality. Figure 2 shows the state diagram corresponding to regular data taking. The same figure also forms the GUI used by operaors.

Transitions between states are accomplished by broadcasting appropriate messages and waiting for acknowlegements. When all relevant acknowledgements are received, the new state is considered valid. A collection of tasks resident on the VME processors are responsible for taking the appropriate actions for a given message. Considerable thought went into choosing the particular communication implementation. For Run II at CDF, the Talarian Smartsockets package was selected. It relies on a publish/subscribe protocol where communications are tagged by subjects



State = IDLE
RunType = Invalid
RunNumber = -1
Transition = (none)
DAQ, Partition = -1
You have control of RunControl

Figure 1: State diagram for datataking and GUI control panel for its implementation. Active elements respond by broadcast of relevant messages to system components.

to which software modules can ”subscribe”. The package is reliable and scales well-an important consideration for a system of 150 crate CPU’s. With bindings in both Java and C, it is matches well with the pre-existing board level C-libraries and host-side Java code. CDF is currently in the integration phase and the structure is meeting the needs of the experiment well.

Figure 3 shows the final design for the Run Control System. Evolution towards the complete system will include the implementation of more sophisticated resource management by Run Control, including coordinating experimental configuration data from a database for experimental operations, a set of packages designed to monitor sampled data from the event stream, and an automated error-handling capability. The monitoring of the event stream is accomplished by C++ processes intimately connected with the ROOT package. The system is described elsewhere in this conference[**?**]. Database needs include access to calibration constants and parameters of the experimental hardware. CDF uses an Oracle database and server. Run Control relies on the jdbc Java facility to make relevant queries to the database and encapsulate data in appropriate Java classes. Some parts of the system (the Silicon Vertex Detector, or SVX) require anomolously large amounts of data. For such cases, Run Control has the ability to delegate database access to dedicated independent processes, or Brokers. It is anticipated that the DAQ system will benefit from the capability to automatically recognize well known error conditions in the system and respond accordingly. A candidate example is localized damage to flip-flops on the SVX front-end chips which manifests itself as a bit in a register turning on or off. Under certain circumstances it is desireable to detect such a condition and reset the system. A highly automated error handler would be useful but a design does not yet exist.

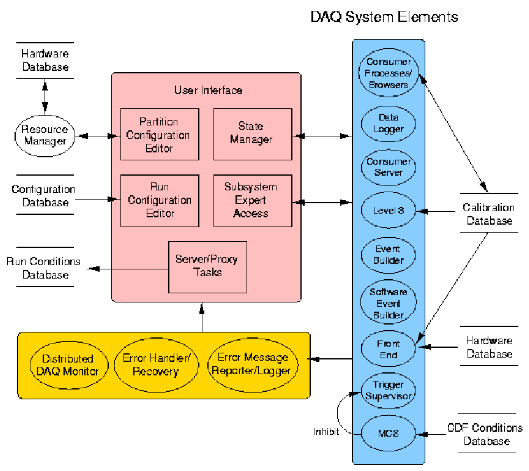


Figure 2: Software components in design of Run Control.

4 Summary

Finally, an aspect of the DAQ that has not yet been dealt with concerns the long-term maintenance of the system. The software throughout the development, integration, and implementation of the system has been designed with an eye towards re-use. The time-span between the first development code and the conclusion of the experiment is a period of nine years, which is much longer than the time-scale for cycles of completely new software technologies to become obsolete or emerge. A related issue is maintaining the expertise to keep the system up and operational. A feature of HEP experiments is the dichotomy between implementing the experimental apparatus and analyzing the results. The attention each receives is often inversley related. This could be much more problematic in the longer scale modern experiments such as CDF.

References

1 H. Wenzel, “, CHEP’00, Padova, Winter, 2000.